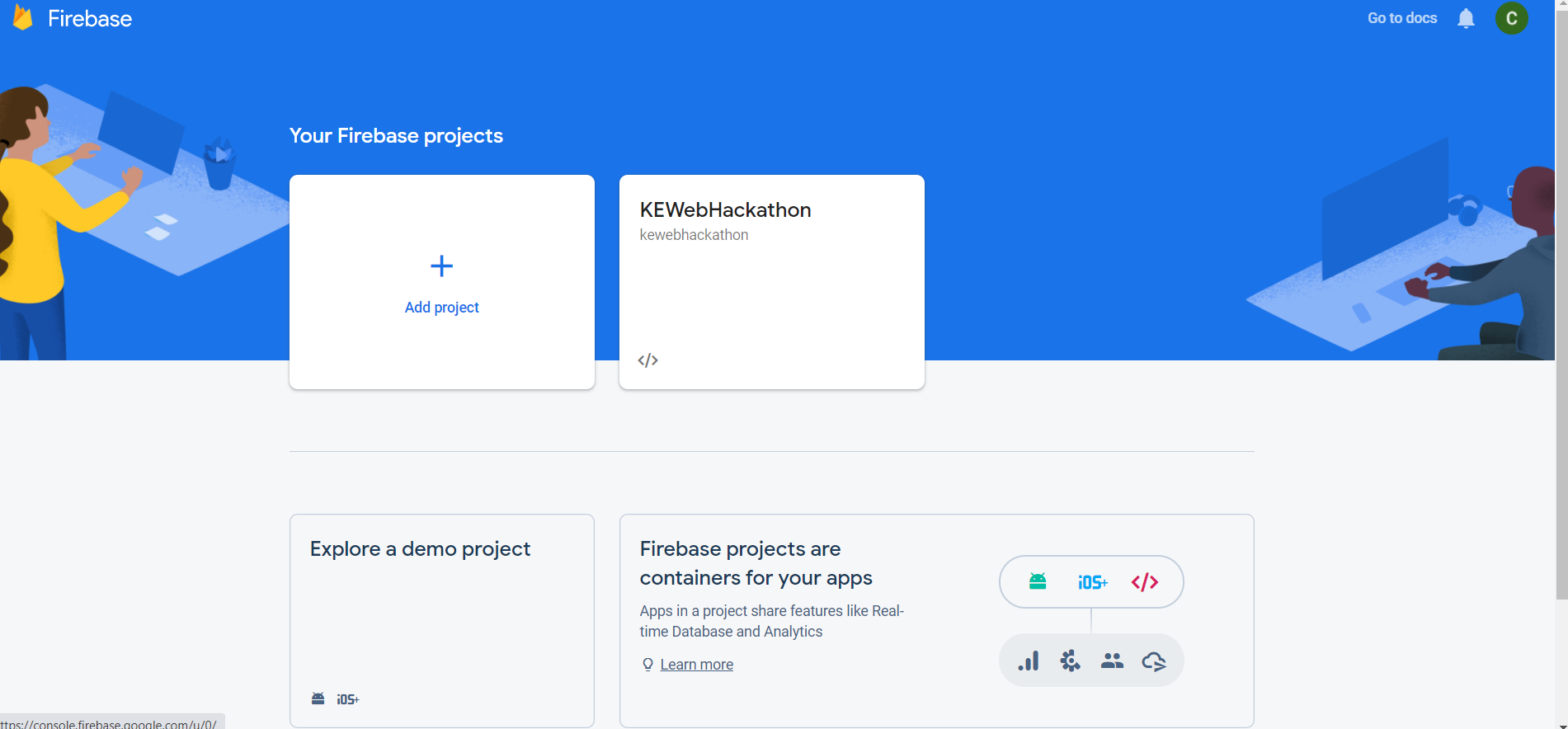
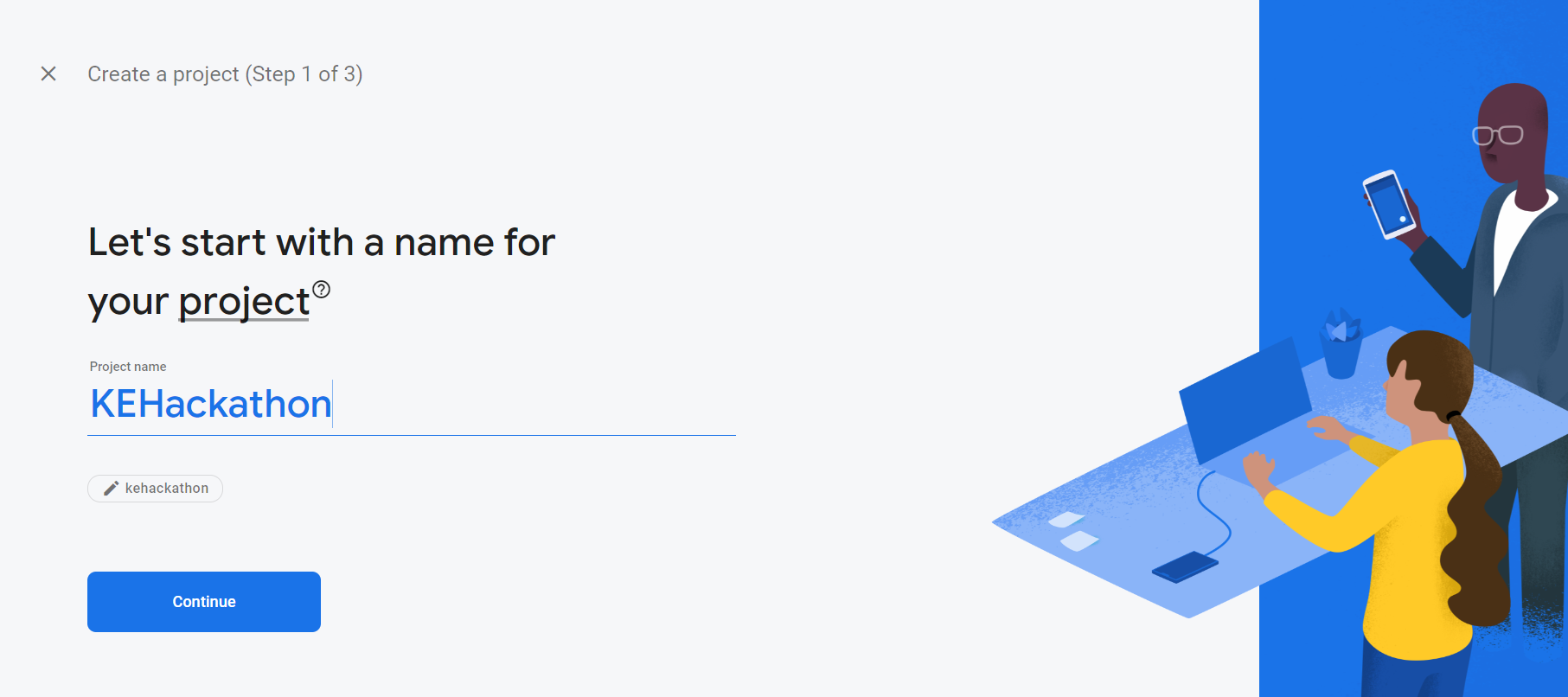
Setting up Firebase Authentication and Firebase Database

1. Firebase Console setup

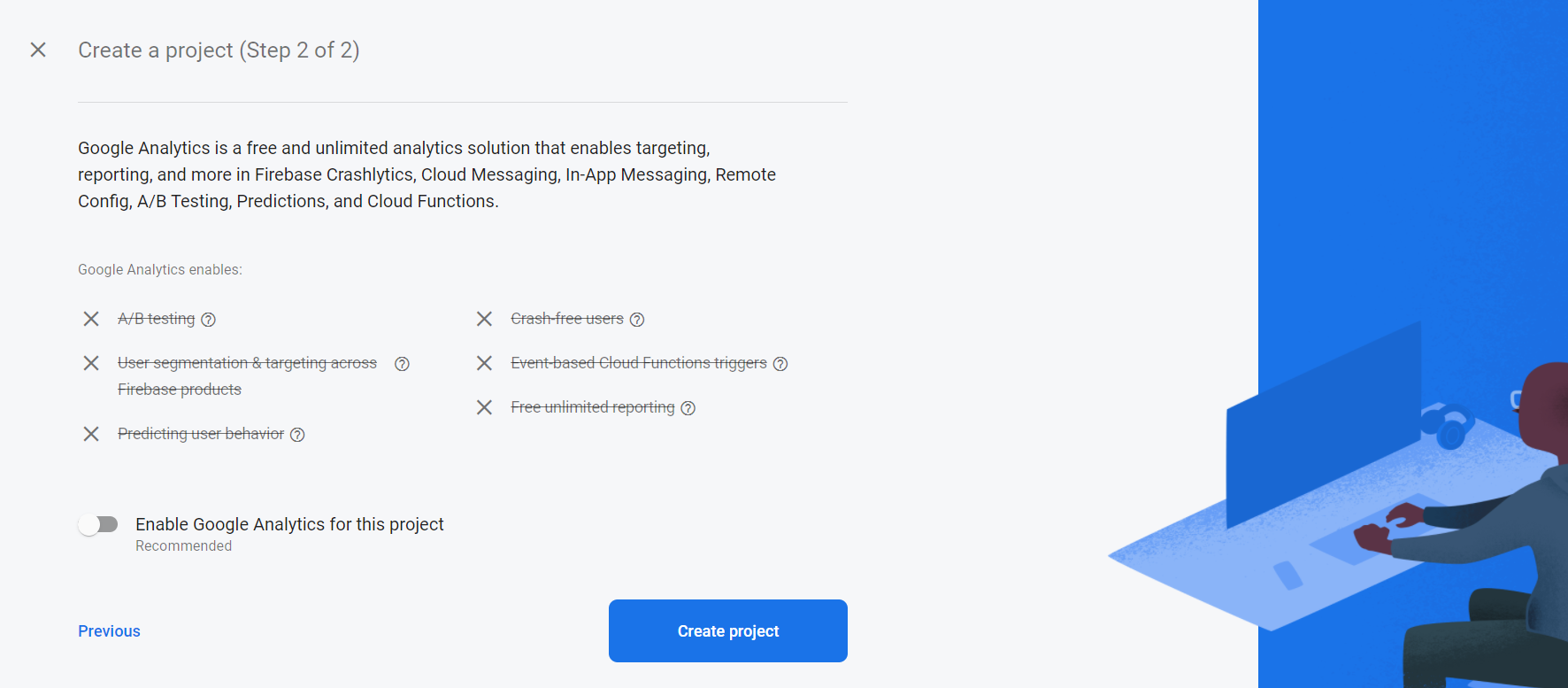
Navigate to URL <https://console.firebase.google.com/> and login with your Google account when required.



Click on Add Project and give the project name. A unique project identifier will be generated (as shown beside the pencil icon)

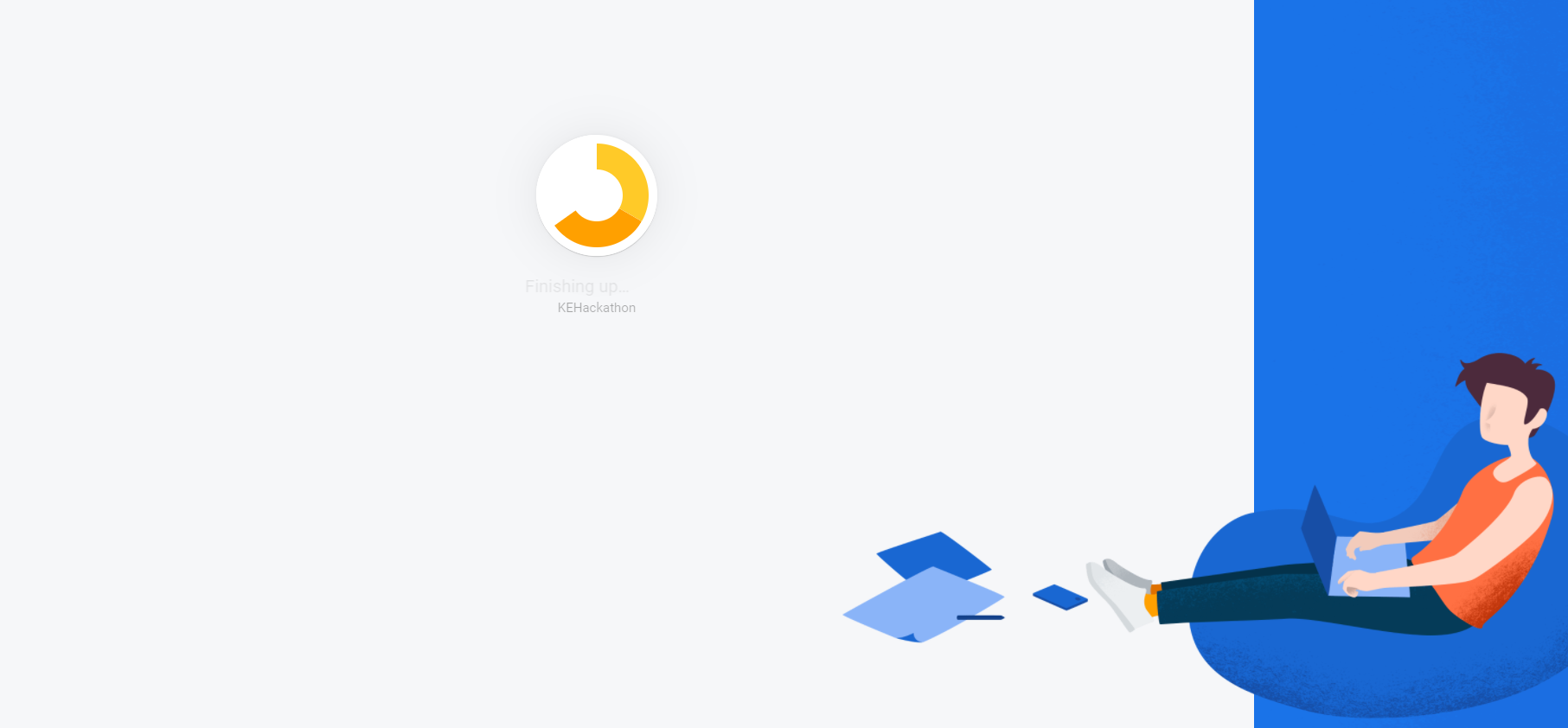


Next, you can choose to either enable Google Analytics or disable it. In this example, we will disable Google Analytics as it is not required.

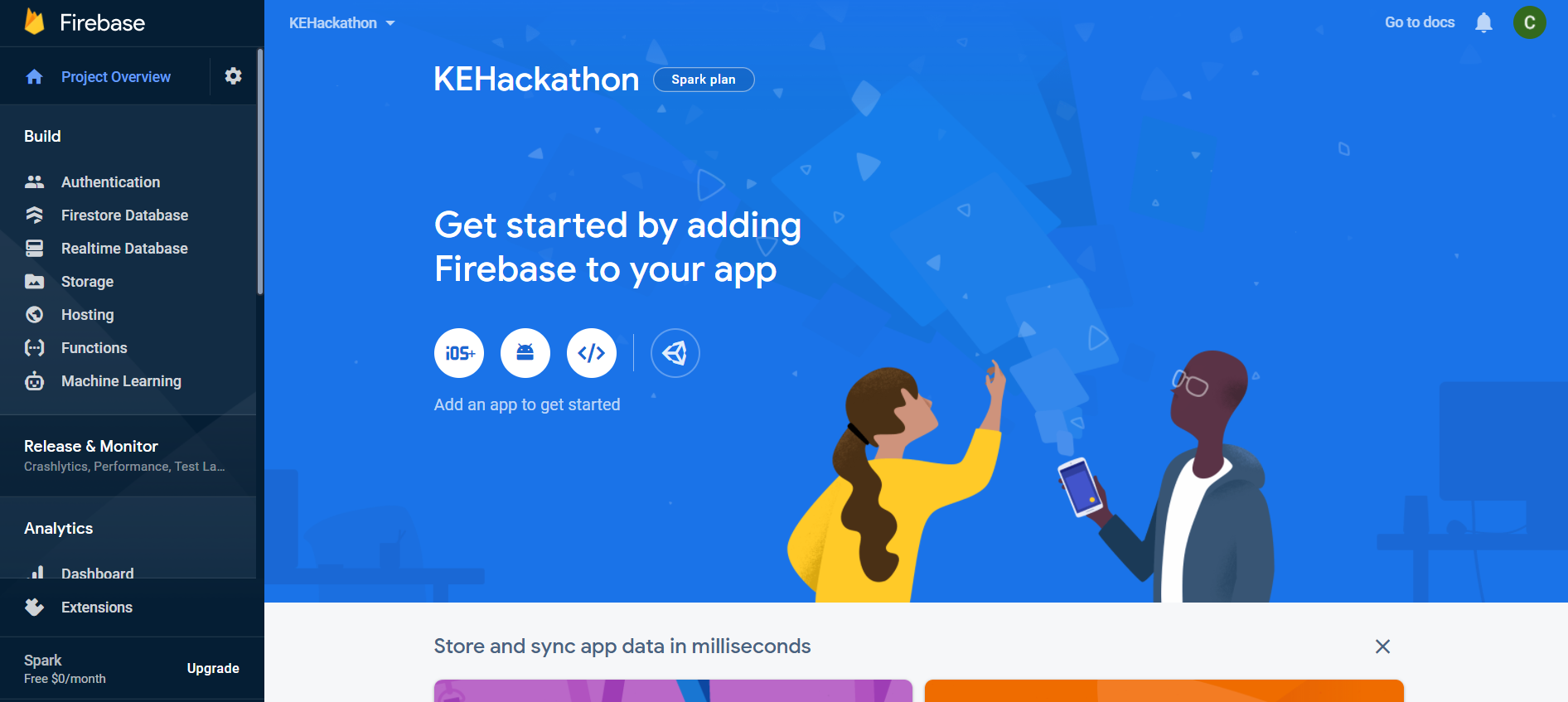


1. Creating the Project

After which, press the Create Project button and wait for the project to be initialised.

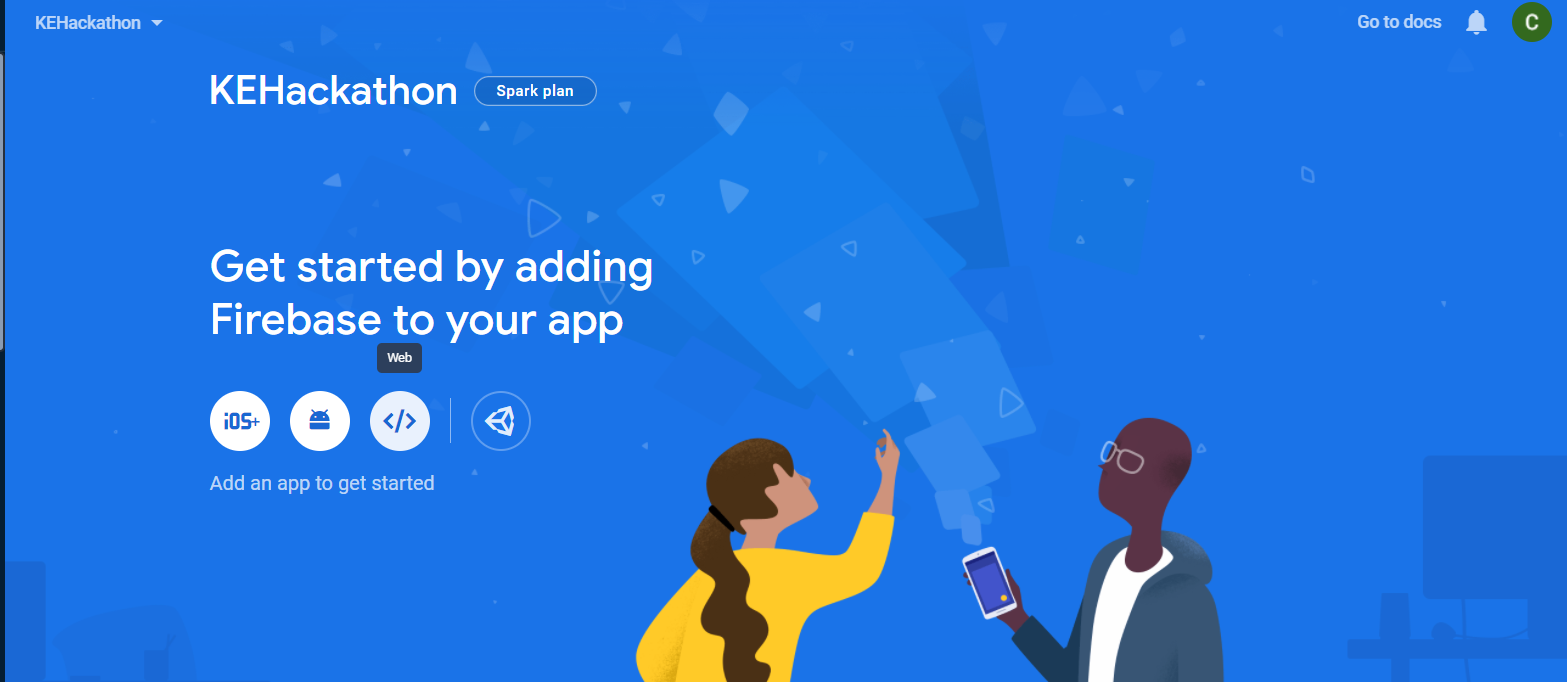


After the project has been initialised, you will be greeted with a dashboard



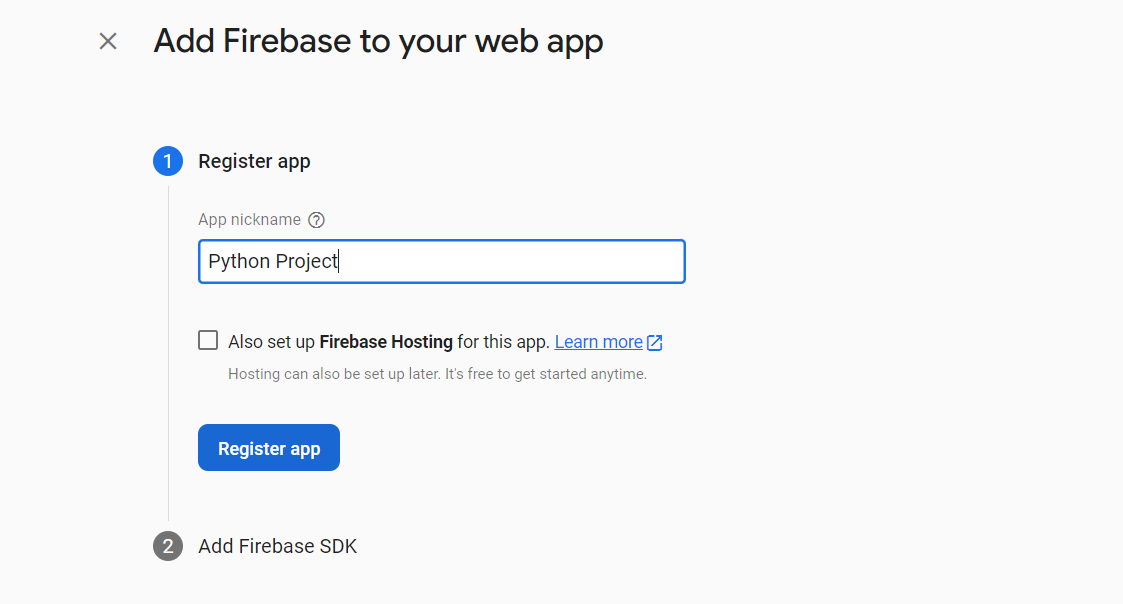
1. Integrating Firebase with your Python project

To get the necessary API keys required, click on the Web button as shown in the screenshot below.

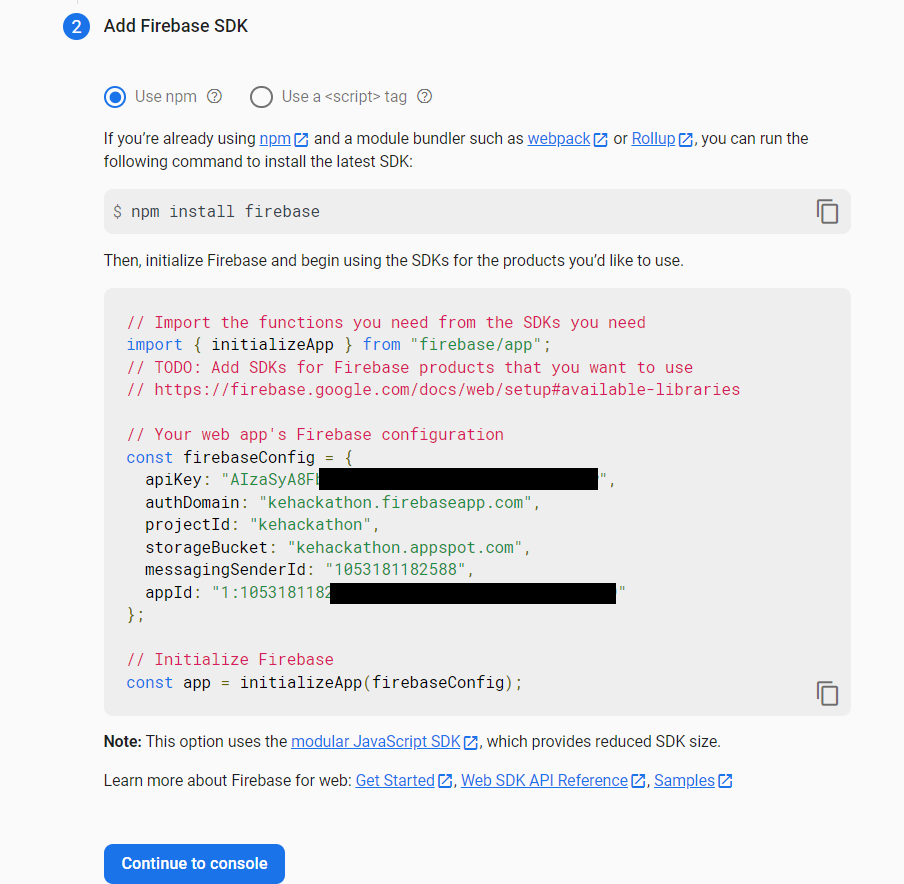


Any name will do for the App nickname as it is only visible to you in the dashboard.

Leave the Firebase Hosting option unchecked and press Register app



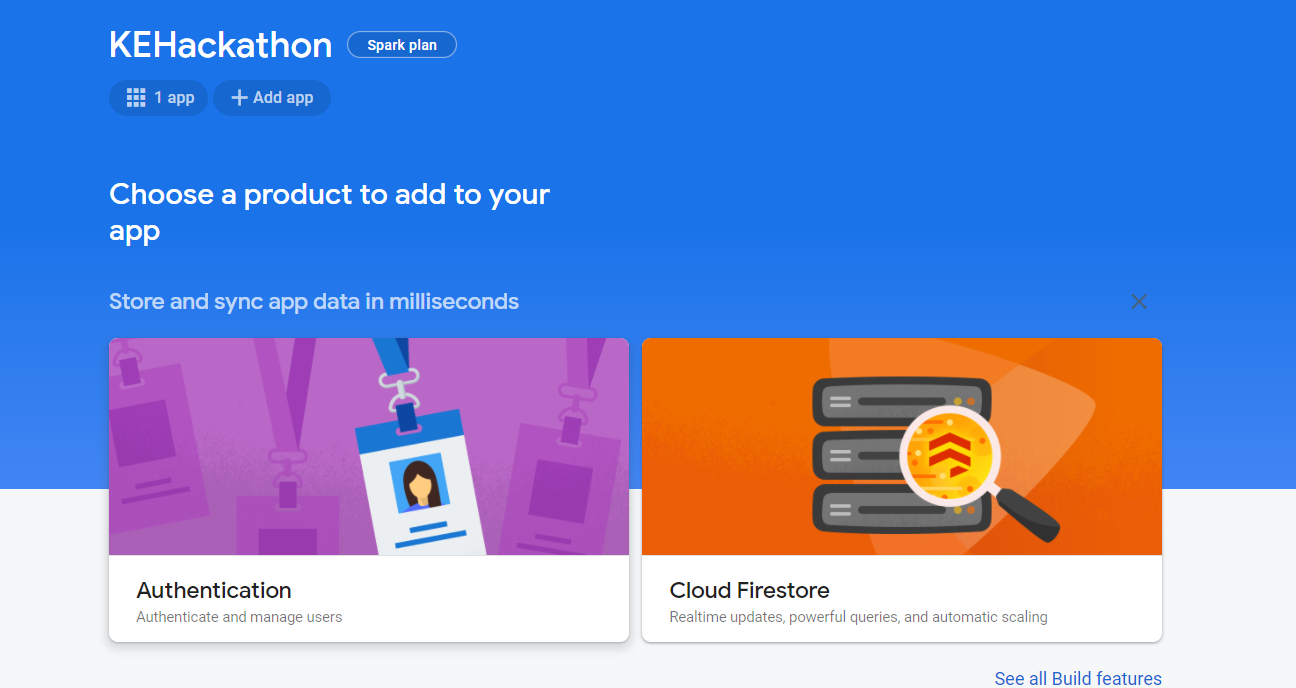
A list of keys will be generated and displayed in the console. These are the necessary keys for your Python code to interact with Firebase. Please do note down the keys and press Continue to console.



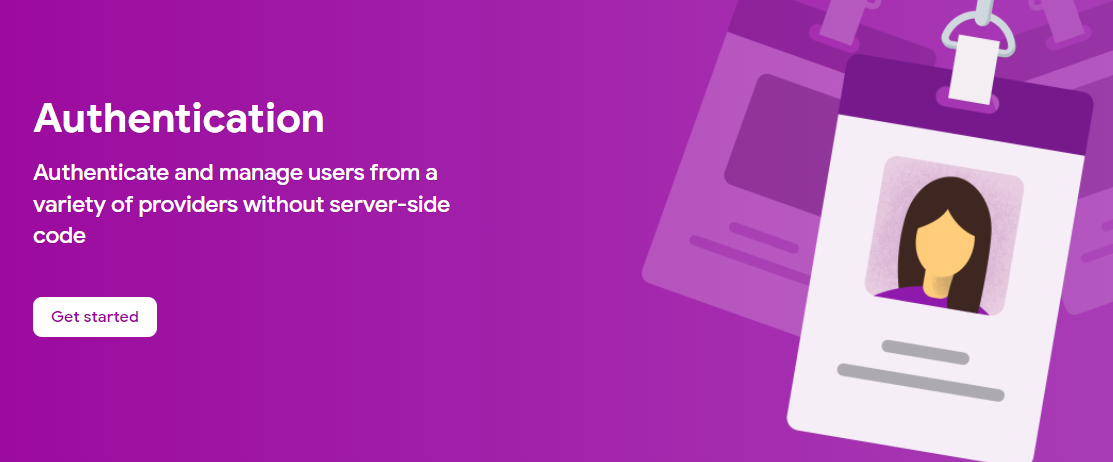
1. Enabling Firebase Authentication

Now that the project has been created, we will enable Authentication for user authentication and create the database required to store the products.

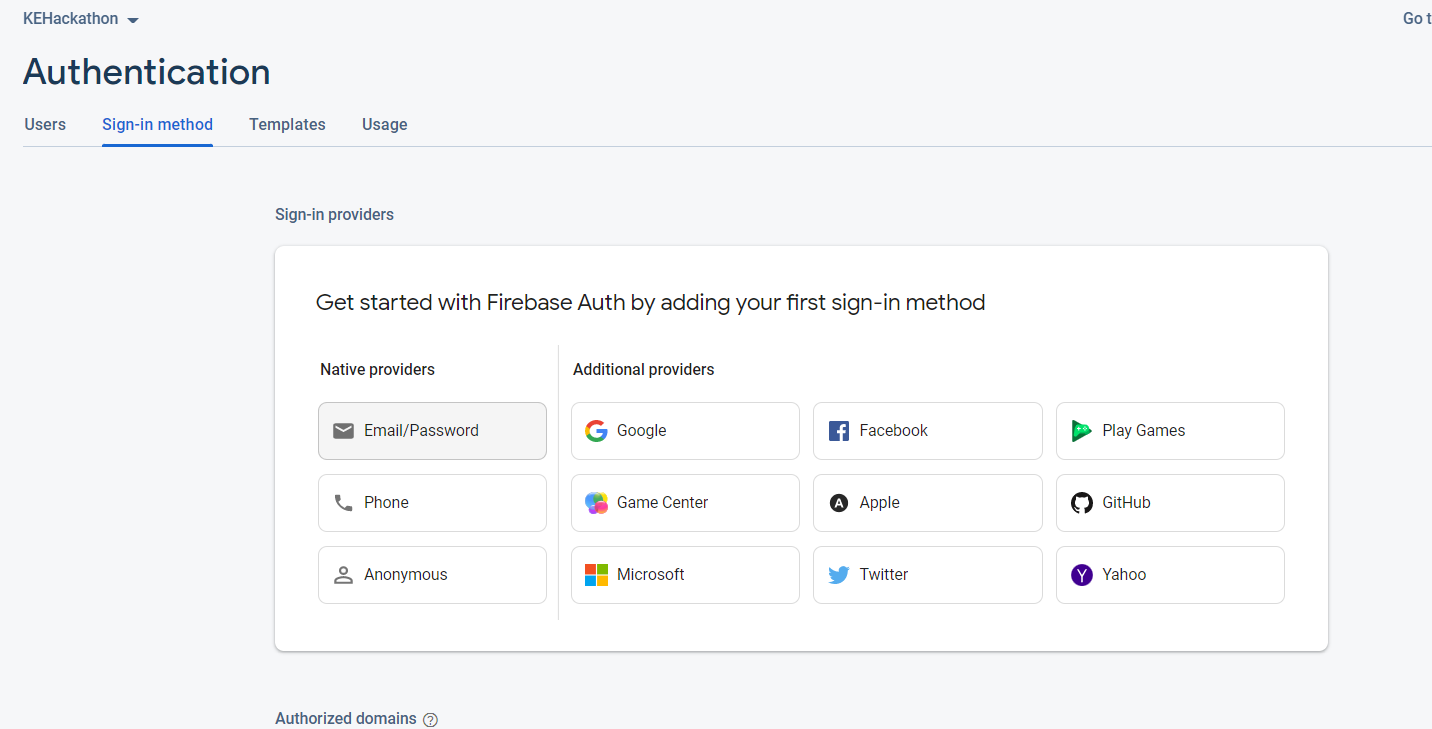
In the console, click on the Authentication button



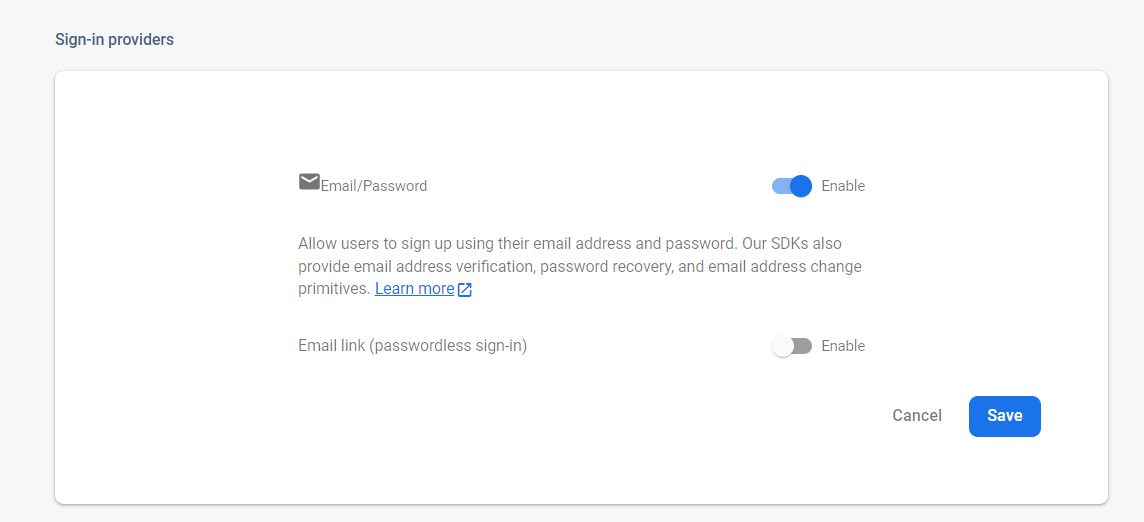
Click on Get Started



Next, click on Email/Password

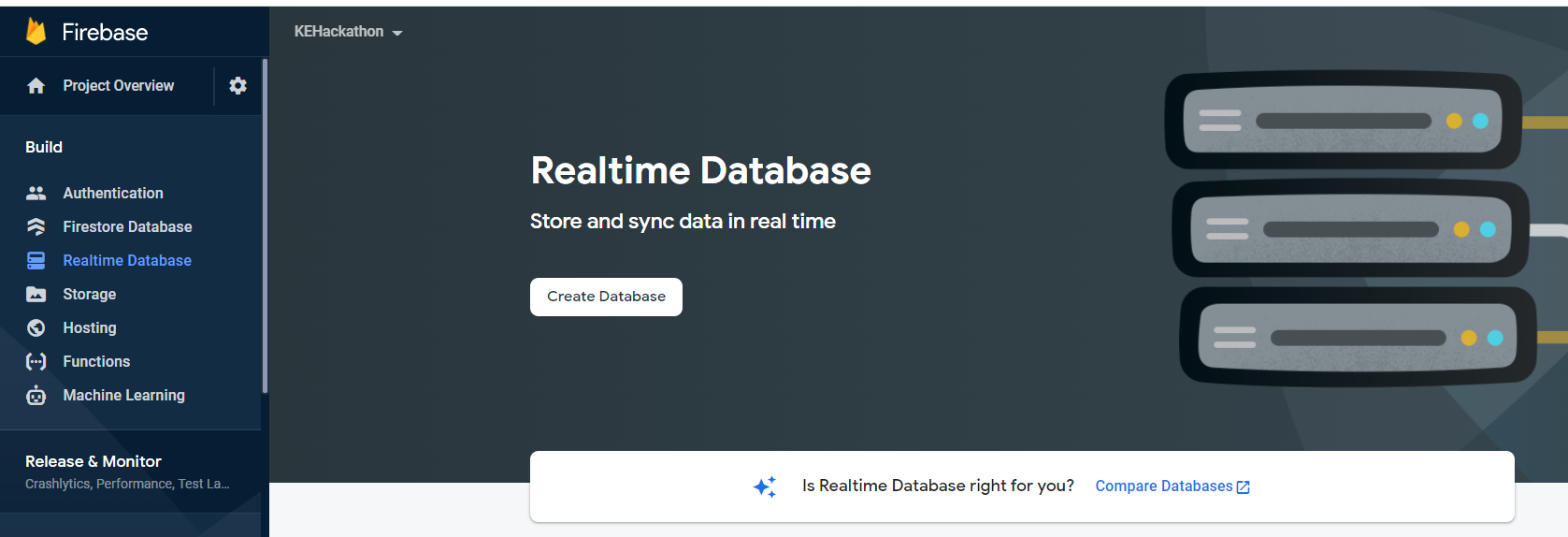


Enable Email/Password and save. The authentication has been setup.

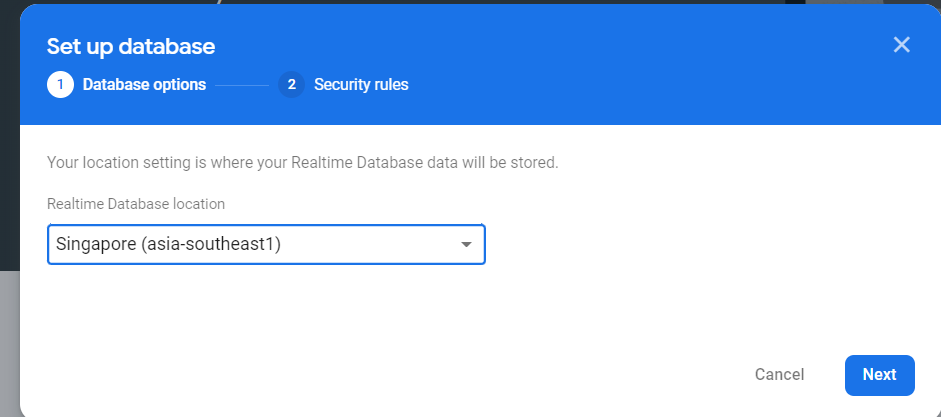


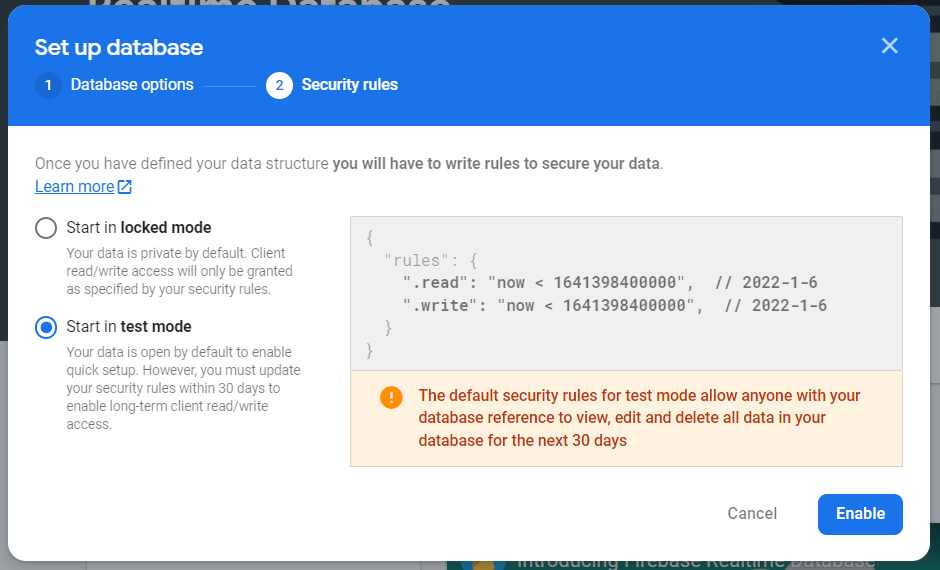
1. Creating Realtime Database

At the sidebar, click on Realtime Database and Create Database

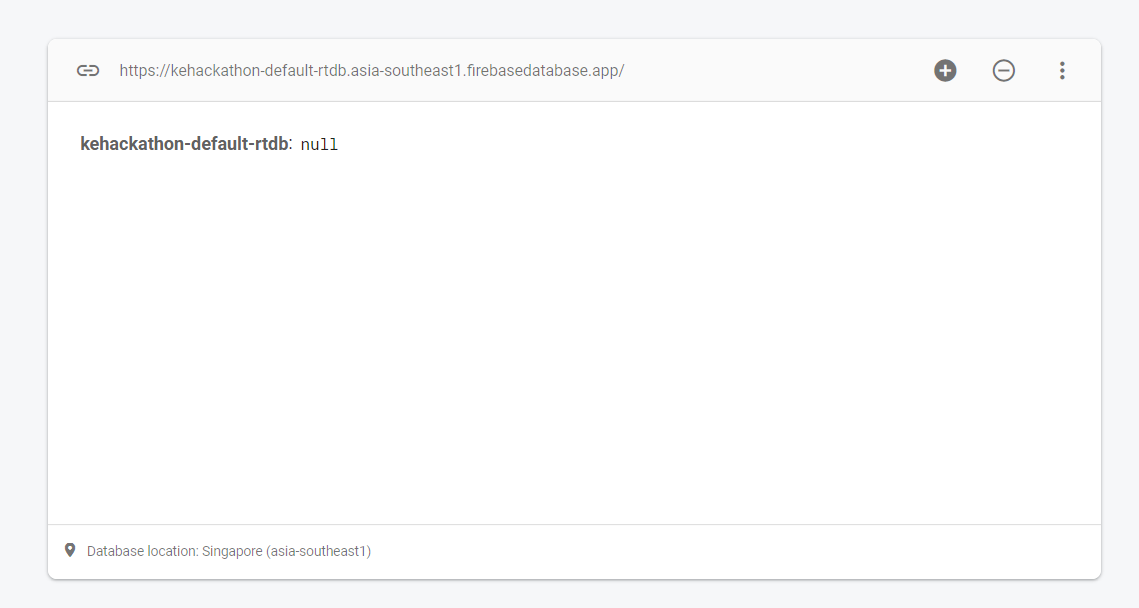


Select Singapore for the location and Test mode for quick access

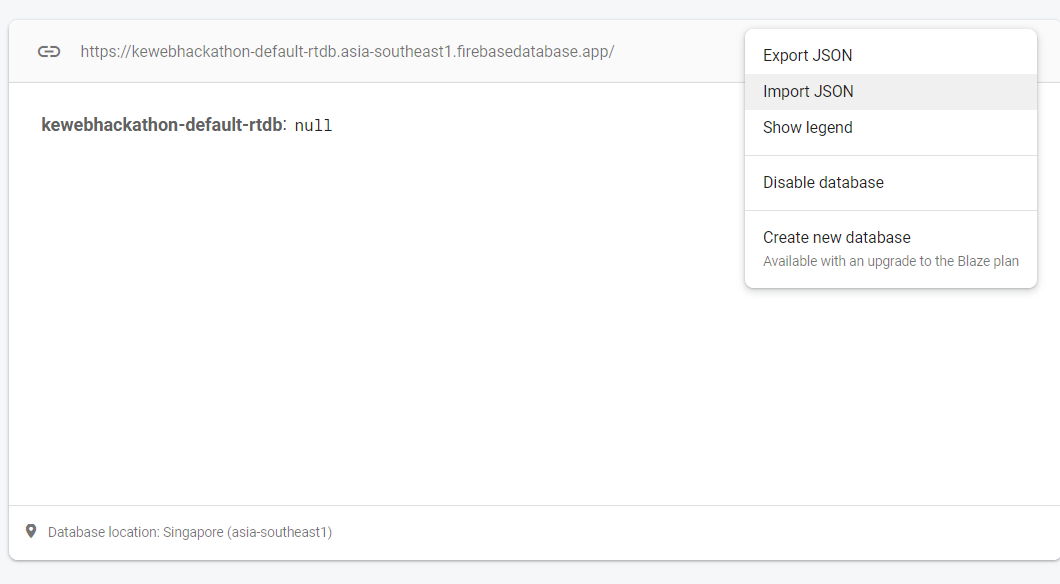




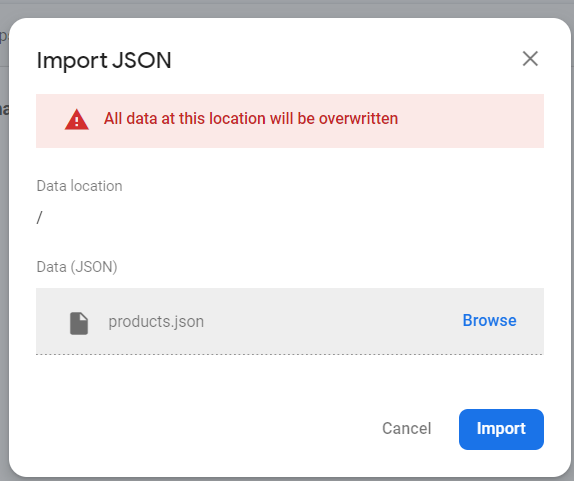
The database has been created. Note down the database URL as shown in the console

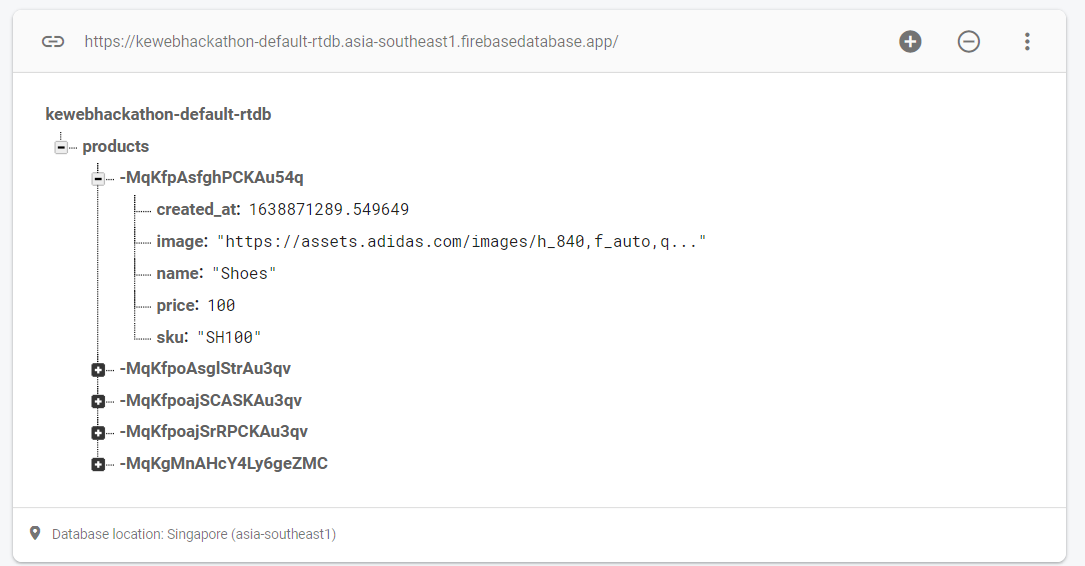


Now, we can insert some data or create our own. Within the code repository, there is a “products.json” file where some sample data are included. Import the JSON by clicking on the additional icon and “Import JSON”

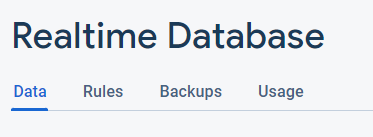


Select the JSON file and import the data.

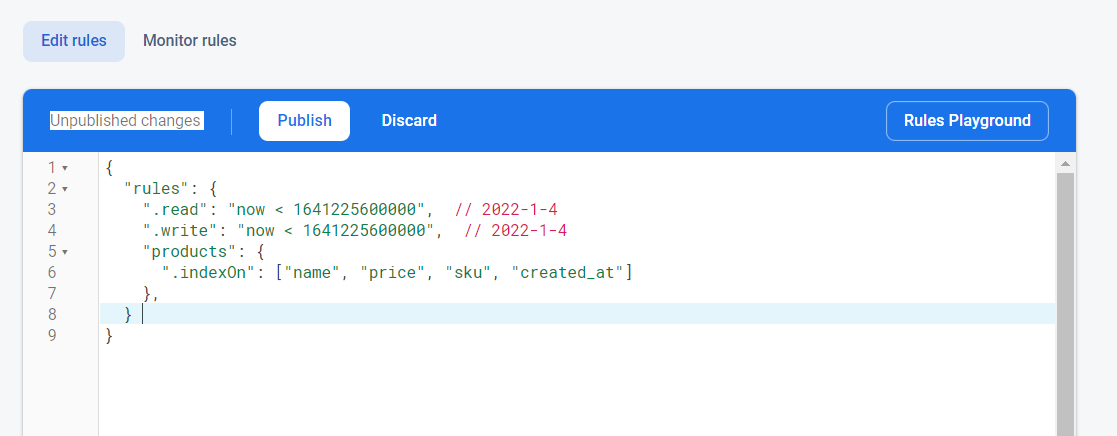




Click on the Rules tab



Copy paste the contents from firebase.txt into the playground and click Publish

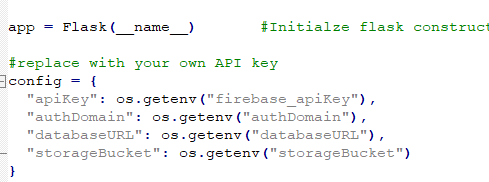


You have now successfully enabled the firewall rules and indexing rules for search

1. Putting all together

Now that we have successfully created the project, it is time to link our Python project to Firebase.

Under the config, replace the “os.getenv(‘firebase\_apiKey’) with the respective keys generated in the earlier steps. If you are proficient in environmental variables, you can create a .env file and do the same requirement.



This will look something like this:



With this, we have successfully completed how to create Firebase Realtime database and enabling Firebase Authentication.

Setting up Typesense Cloud

1. GitHub account setup

Navigate to URL <https://github.com/signup> and sign up for an account by following the instructions on the website. You will need to verify your account using a code sent to your email.

Graphical user interface, text, application

Description automatically generated

Select the free plan when prompted. You may optionally sign up for the additional student benefits by linking your NUS student email, but it is not necessary to do so to setup Typesense Cloud.

A screenshot of a computer

Description automatically generated with medium confidence

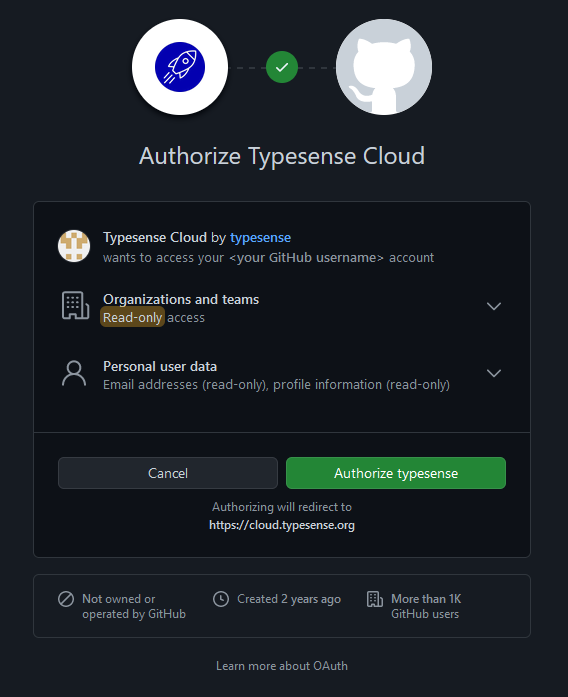
1. Logging In to Typesense Cloud using GitHub

Navigate to URL <https://cloud.typesense.org/> and click the Login with GitHub button

A picture containing chart

Description automatically generated

Click the Authorize Typesense button



1. Creating a new cluster in Typesense Cloud

Next, you will be prompted to create a new cluster. Doing so is free and you do not need to modify any of the default configuration, except for changing the Region to Singapore

Graphical user interface, application

Description automatically generated

Click Launch once you have changed the region for your new cluster. Allow some time for your cluster to be provisioned. It may take slightly longer than the 4-6 minutes indicated for the process to be completed.

Background pattern

Description automatically generated with low confidence

1. Linking Typesense to your Python code

Once your cluster has been provisioned, click on the Generate API Keys button on the top. You will then be prompted to download a .txt file, save it somewhere convenient for you.

Graphical user interface, application, Teams

Description automatically generated

Once the file has finished downloading, open it up with any text editor of your choice and you will be presented with the necessary details to connect to Typesense.

Graphical user interface, text, application, email

Description automatically generated

Under the config, replace the “os.getenv(‘typesense\_api\_key’)” with the Admin API key and “os.getenv(‘typesense\_host’) with the XXXXXXXX.a1.typsense.net domain until the Nodes section from the download .txt file. If you are proficient in environmental variables, you can create a .env file and do the same requirement.

Text

Description automatically generated

Your end result will look something like this

Text

Description automatically generated

With this, we have successfully completed how to create a Typesense account and linking it with our project.